CSCI 103L SPRING 2020

UNIT 0: COURSE INTRODUCTION

WHAT IS COMPUTER SCIENCE

- Should probably be called “Information Science” (or something like that)
  - aside: Information Sciences Institute in Marina Del Rey
- Studying how algorithms can solve or analyze problems
  - Information based problems: given information in some state, analyze or act on the information
  - Computers implement the algorithms
  - Computer is a tool
    - Astronomy is not about telescopes
- CS is NOT programming and programming is not CS
  - Lots of CS is done with math

WHAT DO COMPUTER SCIENTISTS DO?

- Analyze data
- Look for patterns
- Develop algorithms (solutions)
  - From building a website (that works for 500M users/day)
  - To landing a rover on mars with a rocket crane
- Collaborate with other scientists
  - Most (all?) other science requires computers these days
    - Physics, chemistry, biology, medicine, even math
- Program

THIS COURSE: CS103L INTRODUCTION TO PROGRAMMING

- INTRODUCTION
  - No need for prior programming experience
  - >90% have programmed
  - Course moves quickly: novices ~~may~~ *will* need to put in significant time
- PROGRAMMING
  - The course teaches the basic programming constructs and techniques
- 103L
  - L = has a LAB. You will practice the art of programming and demonstrate your (new) abilities

TEXT

RIGHT PLACE FOR ME?

- 102 > 103L > 104L
- 103L (4 units): fairly quick pace, focuses on syntax of C++, focuses a lot on memory, pointers, stack
- 104L (4 units): data structures in C++, object oriented programming
- 102 (2 units): “on-ramp” to programming. Programming basics, computational thinking.

TEXT

102 VS. 103L

- Never programmed before
- 103L seems like it might be too fast
- Want measured introduction to programming and C++
- Contact CS department and/or Viterbi advisors for more information.

TEXT

103L VS 104L

- Good programming experience
- Understand idea of algorithms and run-time (complexity), lists, graphs & trees
- Some object oriented design/programming
- Possibly some C++
- You can take the 103L challenge exam
  - Contact Ryan Rozen <rozan@usc.edu> and he'll administer the challenge exam
  - csdept@usc.edu

WHAT IS PROGRAMMING?

- Programming, code, coding, coder…
- High level:
  - Come up with a way to solve a problem (algorithm) ← this is CS
  - Tell a computer how to do it ← this is programming
  - In this course we often do both
    - 170/270 usually algorithm only

. . . "COMPUTER. . . "

- Wouldn't it be great if...
  - "Computer... turn on the porch light at 8pm."
    - Oh, wait...
  - "Computer, analyze the stock market using a Laplacian differential and identify the best stocks for today..."
    - Star Trek computer *isn't* quite there yet

WHAT DO YOU ACTUALLY *DO* TO "PROGRAM"

- Details depend on programming language - high level is the same
- Big Picture: computational thinking, decomposition, abstraction

TEXT

COMPUTATIONAL THINKING

- Thinking like a computer
- Understanding how to represent a problem in a way computers can solve
- There are other definitions along the lines of logical problem solving

TEXT

DECOMPOSITION

- Breaking a big problem down into smaller pieces
  - Solve (implement) each piece
  - Combine to solve big problem
- Also allows reuse, isolation, encapsulation
  - Key software engineering concepts

TEXT

ABSTRACTION

- Reducing or distilling a problem or concept to the essential qualities
  - Simple set of characteristics that are most relevant to the problem
- Many (most, all) of what we do in engineering and computer science involves abstractions

WHAT DO COMPUTERS DO?

- Tell me what computers do?
- Computers do two things:
  - Discreet math on binary numbers - really, really stupidly fast
  - Move binary numbers from one place to another (sometimes fast, sometimes slow)
- Thats it.
  - No really, that’s it.

TEXT

BACK TO PROGRAMMING

- Programming is the art of taking an algorithm and implementing it on a computer
- But how?
- Need a programming language
- Programming language has:
  - A way to represent data and steps in a human readable form so an actual human can write a program (usually text)
  - A way to turn that into binary numbers that actually make the computer do something
    - Remember: computers only understand binary numbers
    - This is usually a compiler or interpreter

TEXT

PROGRAMMING LANGUAGE LEVEL

- Programming languages come in levels (of hell)
  - Very low level - binary hacking, firmware tweaks
  - Assembly (using basic operations of CPU)
  - Compiled (C,C++, Go, Rust) ← this course
  - Interpreted (Python, Ruby, JS) ← lots of “the web” done here
  - Scripting (bash, etc. Make other jobs easier)
  - Graphical (flow based, Scratch, LabView)

TEXT

THIS COURSE: C/C++

- Why C/C++?
  - Widely used in industry (for a number of reasons)
  - Low-level in some way
    - Very literal memory model
  - High-level in other ways
    - Object oriented
  - C is common in embedded hardware (IoT)
    - For you CECS majors
  - Used to implement Operating Systems (Linux, macOS, Windows)
  - Used to implement *other* languages: Python, CRuby, MATLAB

TEXT

IS C/C++ RIGHT FOR INTRO CS COURSES?

- In other words why not Java (Python)?
- Answer: bumper bowling.

SYLLABUS

- Review the Syllabus…
- And take a look at bytes.usc.edu

TEXT

COURSE ADVICE

- Over estimate the amount of time it will take
  - By a lot some times
  - You can spend *hours* tracking down a subtle bug
- Avail yourself of resources
  - CP/TA/Prof office hours: don't be embarrassed. Lots of people will have questions, struggles, etc. Also VARC tutoring, online resources, etc.
  - Peers: yes, but don't get caught up in an academic honesty violation
    - Getting a zero on one PA = lower final grade by half scale (C → C-)

TEXT

COURSE ADVICE #2

- Experiment: you will write programs that suck (or simply don't work). Rewrite them and learn
- Practice, practice, practice
  - Programming is a SKILL
  - Some people "get it" and won't struggle
  - Some people take time to understand programming

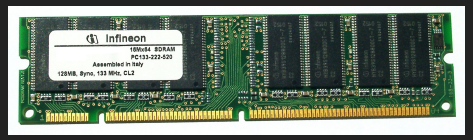
TEXT

WHO IS THIS PROF. GOODNEY GUY?

- Who Am I?
- Teaching faculty in CS
- Undergrad at USC in EE
- Masters at USC in EE
- PhD at USC in CS
- Worked primarily in visual effects & motion picture industry (~15 years exp.)

COMPUTER MODEL

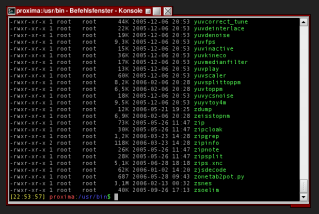
- How do we *think* about the computer we're programming



Does math, moves data

TERMINAL

Interactive I/O with a human

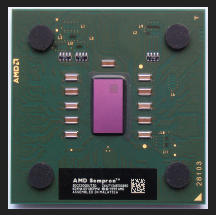


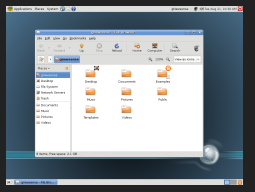
MEMORY

CPU

INPUT/
OUTPUT

Lots of binary numbers





FILES

I/O to files on disk

TEXT

COMPUTER MODEL IN THIS COURSE

- Everything we do in this course are operations on memory
  - Set memory to a value
  - Move memory from one location to another
  - Read input and put in memory
  - Read from memory and output to terminal

MEMORY

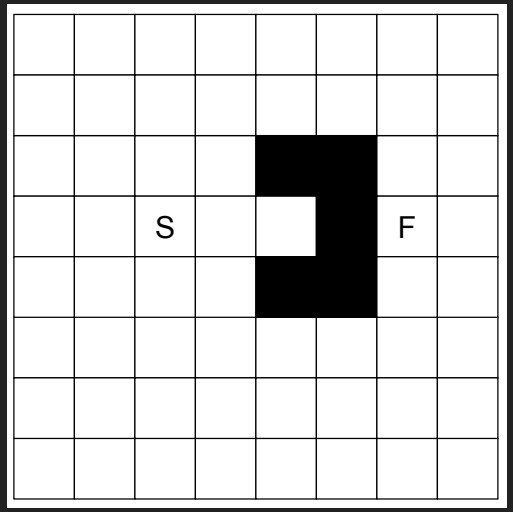
- Memory is made up of locations
- Locations hold data
- Each location has an address
- Data size can vary
- Memory has two operations
  - Write: set memory at address X to Y
  - Read: return data at address X
- Granularity of address is 1 byte = 8 bits

Address	Data
0x0000000000000000	32
0x0000000000000008	1
0x0000000000000010	342124
0x0000000000000018	‘a’
0x0000000000000020	3.141592
0x0000000000000028	
0x0000000000000030	
...	
0x0000000FFFFFFFE8	
0x0000000FFFFFFFF0	
0x0000000FFFFFFFF8	

TEXT

COMPUTATIONAL THINKING: PATH PLANNING

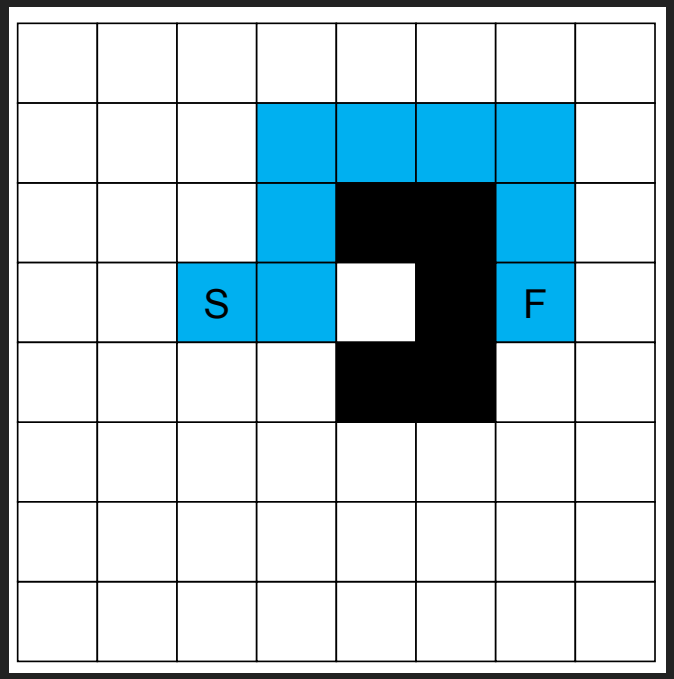
- Lets look at another task: path planning
  - Here its a maze, but the same problem/algorithms used by Google Maps, etc.
  - Task: find shortest path from S to F.



TEXT

COMPUTATIONAL THINKING: PATH PLANNING #2

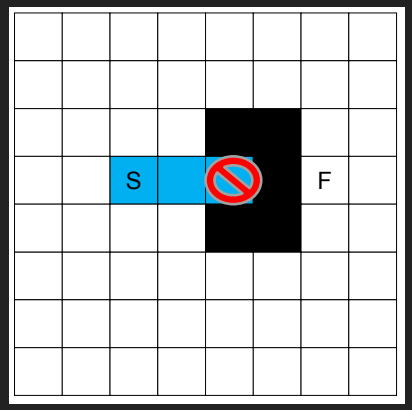
- Jump to the answer.
- Easy for humans to do.
- Can you think like a computer?
- Can we come up with some strategies?
  - Is there more than one algorithm?

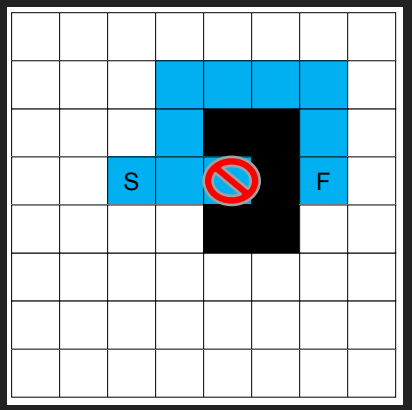


TEXT

COMPUTATIONAL THINKING: PATH PLANNING #4

- This still looks global…
  - Not a realistic scenario
- What is the algorithm?

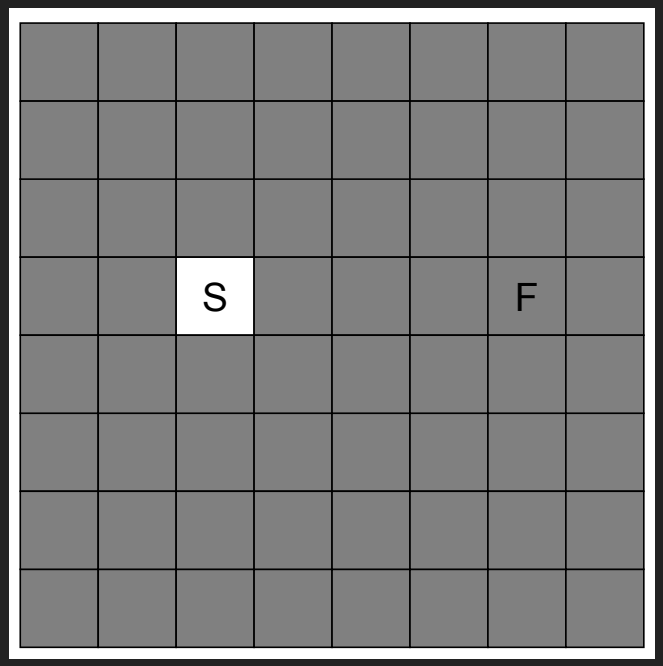




TEXT

COMPUTATIONAL THINKING: PATH PLANNING #3

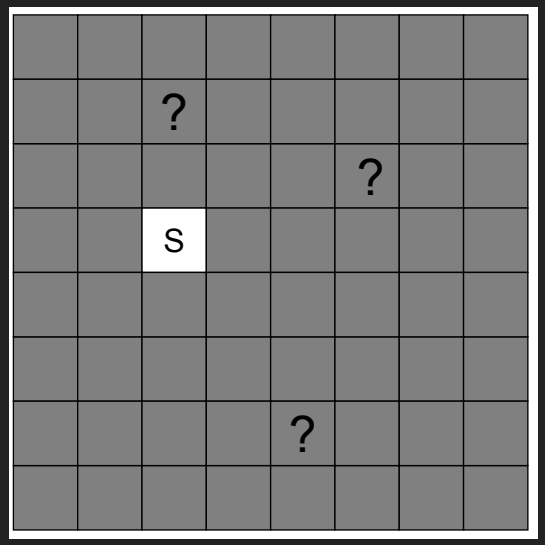
- What if it looked like this?
  - Obstacle (wall) is hidden?
  - Does that change your strategy?
- Local vs. Global view



TEXT

COMPUTATIONAL THINKING: PATH PLANNING #5

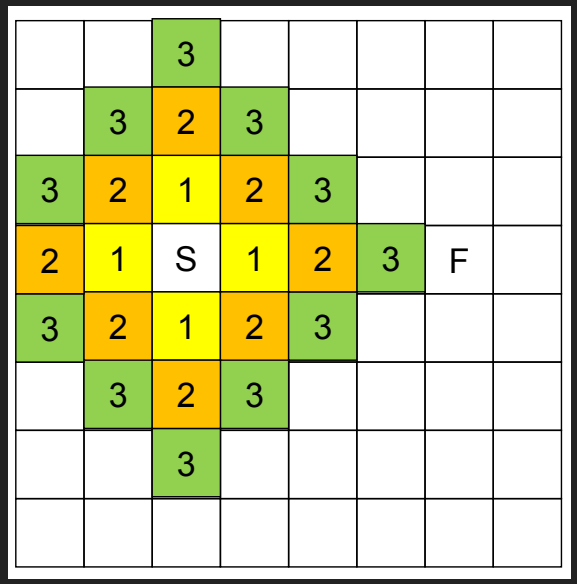
- Given a starting square and a number of hidden obstacles and a hidden finish square, develop a generic algorithm that finds the shortest path.
- Obstacles and the finish can be observed from adjacent squares.



TEXT

COMPUTATIONAL THINKING: PATH PLANNING #6

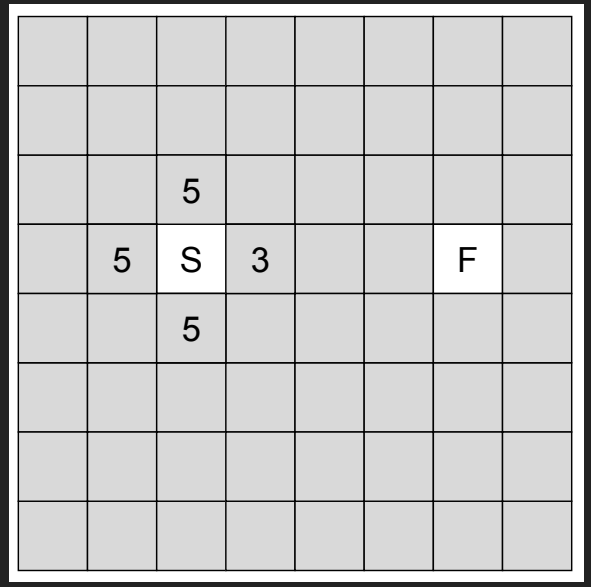
- Solution: examine all adjacent squares from the start. Note distance from start and if the square is an obstacle or if it is the finish. When done, pick one of the newly examined squares and examine it's neighbors. Repeat until the finish is found (or you run out of squares)
- Breadth first search
  - PA4 (probably)



TEXT

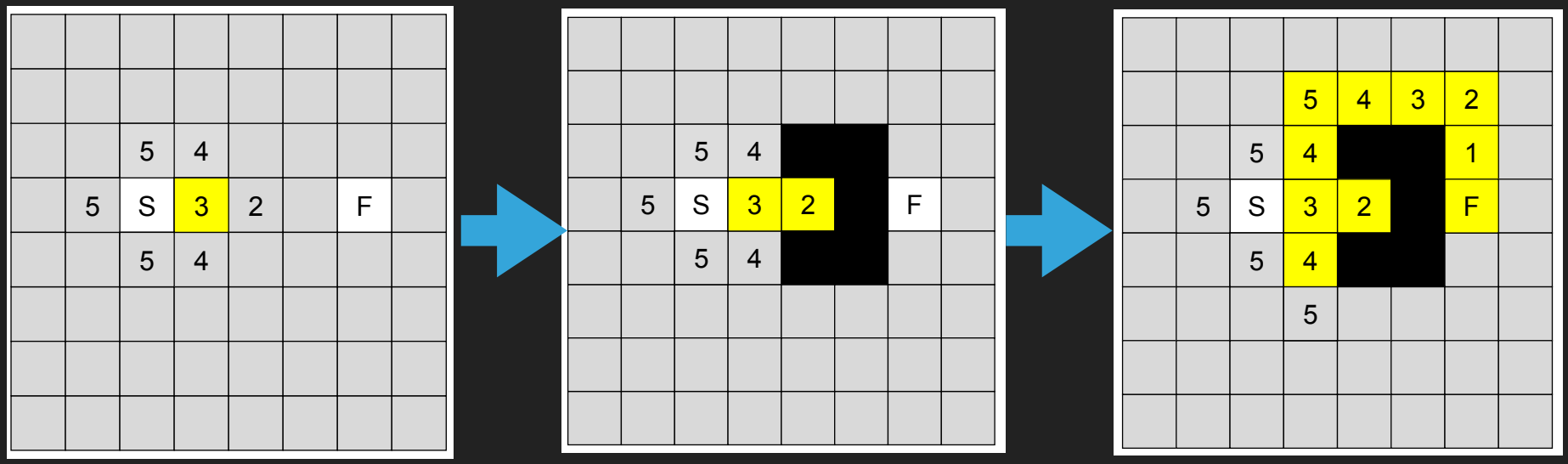
COMPUTATIONAL THINKING: PATH PLANNING #7

- What if we tell you where the finish line is? Does this change the algorithm? Does it change what data we're keeping track of (our abstractions)?
- Before we explored all unexplored squares, in order.
- Now we explore the square with the minimum distance.



COMPUTATIONAL THINKING: PATH PLANNING #8

- Heuristic Search



TEXT

ALGORITHMS VS. PROGRAMMING

- What we just did was algorithm development
- What do we need to do to program it?
- What do we need to keep track of?
- What abstractions do we need?
  - “Real World” things that need to be in our program?
- Steps to solve the problem?

COURSE MECHANICS

- To program you need:
  - Text editor/Development environment
  - Compiler
  - Debugger

TEXT EDITOR

- Many, many choices
  - Sublime, Atom, Notepad++, gedit, Xcode, VSCode, emacs, VI, nano
- Lots of features to look for, but most do a good job
  - Find one that you like, very much a personal preference
- Vocareum (more in a sec) has an editor, but you are free to edit/run/debug locally
  - We encourage you to install a development environment locally on your laptop as a learning experience

TEXT

COMPILER

- Takes C/C++ source code, generates machine code
- We use clang (Apple open source project) and/or GCC (GNU compiler collection)
- Vocareum has both

TEXT

DEBUGGER

- All code ends up with bugs
- Debugger helps you “see” the code while its running
  - Single-step, step-in, step-out, examine variables and data
- We provide gdb and lldb, both are good
  - Some environments like Xcode have GUI interfaces to debuggers

TEXT

VOCAREUM

- https://labs.vocareum.com
- Web-based programming environment
- You'll be automatically enrolled the first week of class
- All labs and programming assignments are done on Vocareum

TEXT

FAQ

- Most F of the FAQ at this point: can I use XYZ to code?
  - Answer: yes, please do.
  - Vocareum makes access easy. Any computer with a web-browser can access the programming environment
- I don’t have a laptop?
  - Any of the lab computers can access the programming environment

TEXT

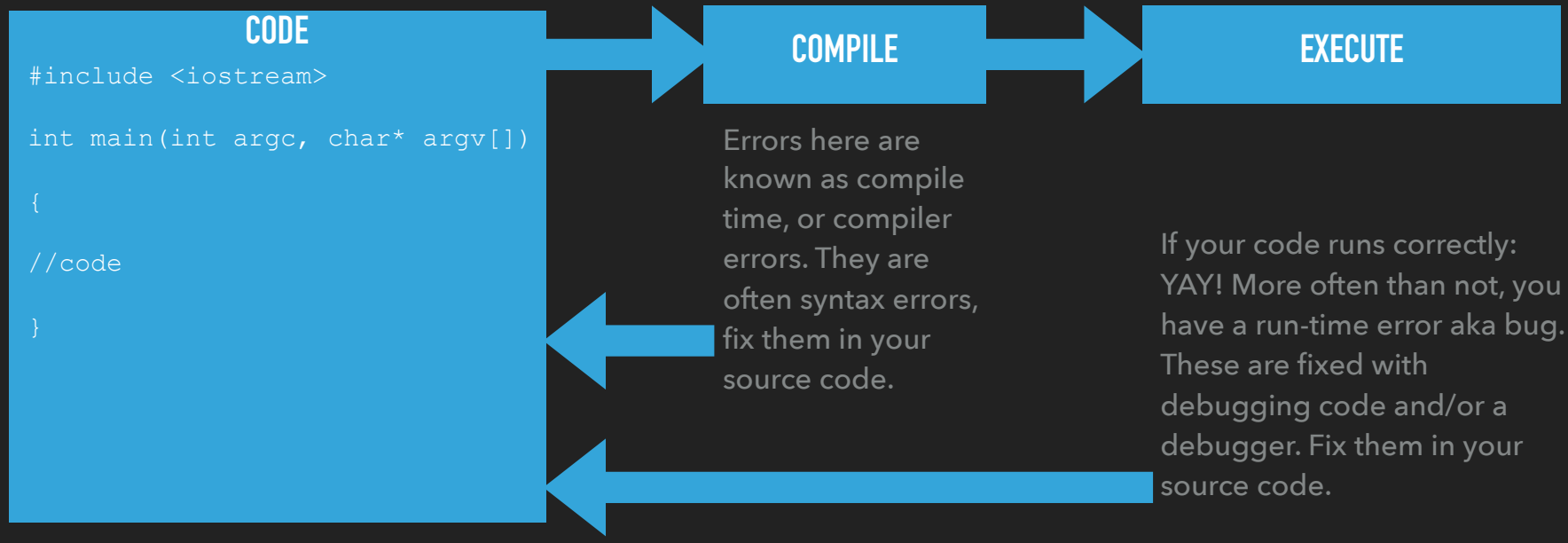
C/C++ PROGRAM

- What does C/C++ code look like?
- Comments
  - /* … */
  - //
- Variables, functions, statements
- main()
  - This is where the program starts

```
/* Anything between slash-star and
 star-slash is ignored even across
 multiple lines of text or code */
/*----Section 1: Compiler Directives —-*/
#include <iostream>
#include <cmath>
using namespace std;
/*------------ Section 2 ----------------*/
/*Global variables & Function Prototypes */
int x=5; // Anything after "//" is ignored
void other_unused_function();
/*----Section 3: Function Definitions ---*/
void other_unused_function()
{
 cout << "No one uses me!" << endl;
}
int main(int argc, char *argv[])
{ // anything inside these brackets is
 // part of the main function
 int y; // a variable declaration stmt
 y = x+1; // an assignment stmt
 cout << y << endl;
 return 0;
}
```

TEXT

C/C++ DEVELOPMENT CYCLE



TEXT

C/C++ DEVELOPMENT CYCLE

- Get comfortable with the code/compile/run/debug loop
  - You will do it A LOT. Sometimes 100's of iterations per assignment.

TEXT

C/C++ CODING TIPS

- Take small bites!
- Don’t try to code the whole assignment - you will have lots of compile time errors to wade through before you even get to the run-time errors
- Try to implement one step of your plan at a time
  - Often you can code and test one step of your algorithm at a time
- Have Fun!

TEXT

DEMO ON VOCAREUM

- Launch Vocareum
  - Also show local on my laptop
  - And https://www.onlinegdb.com/
- Write some code
- Compile
- Run
- Yay!

TEXT

ACKNOWLDGEMENTS

- All images and graphics in this slide deck courtesy of Wikimedia Commons unless otherwise noted
- Maze graphics courtesy of Mark Redekopp